UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/050,358	01/16/2002	Nicolai Kosche	004-7047	5734

42714 7590 03/22/2006

ZAGORIN O'BRIEN GRAHAM LLP (004)
7600B NORTH CAPITAL OF TEXAS HIGHWAY
SUITE 350
AUSTIN, TX 78731-1191

EXAMINER	
WANG, RONGFA PHILIP	

ART UNIT	PAPER NUMBER
2191	

DATE MAILED: 03/22/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/050,358	KOSCHE ET AL.	
	Examiner	Art Unit	
	Philip Wang	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *03 January 2006*.

2a)☒ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-61* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-61* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *03 January 2006* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date 2/14/03, 4/14/03, 2/10/06
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This office action is in response to amendment filed on 1/3/2006.

2. The objection to the specification is withdrawn in view of the Applicant's amendment to specification.

3. The objection to the abstract is withdrawn in view of the Applicant's amendment to abstract.

4. The objection to the drawing is withdrawn in view of the Applicant's submission of replacement sheets.

5. The objection to the claims 8, 22, and 33 is withdrawn in view of the Applicant's amendment to the claims.

6. The 35 U.S.C § 101 rejections of claims 1-7, 9-25, 28-49 and 59-61 is withdrawn in view of the Applicant's amendment.

7. The 35 U.S.C § 112 first paragraph rejections of claims 3, 30, 45, 49 and 60 is withdrawn in view of the Applicant's persuasive argument.

8. The 35 U.S.C § 112 second paragraph rejections of claims 1, 19, 47, 54 and 58 is withdrawn in view of the Applicant's amendment.

9. Per Applicant's request, claims 1, 3, 8, 9, 19-22, 25, 32, 33, 39, 46, 47, 50, 54, 55, 58, 59 and 61 have been amended.

10. Claims 1-61 remain pending.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

1. Claims 1-11, 12-15, 19-49, 55-60 are rejected under 35 U.S.C. 102(b) as being anticipated by Anderson et al. (US Patent No. 5,964,867).

As per claim 1, Anderson et al. disclose

executing the code on a processor (col. 26, line 61, "Machine code is executed..");

detecting the execution event (col. 5, line 39-40, "... sampling the program counter when event counters overflow...");

and backtracking from a detection point in the code coinciding with the detection of the execution event to a preceding operation associated with the execution event, the backtracking identifying the preceding operation at a predetermined displacement from the detection point unless an ambiguity creating location is disposed therebetween, wherein the predetermined displacement is dependent upon the preceding operation which is indicated by the execution event (col. 1, line 35-40, "... to monitor specific events during execution of a program..."; col. 3, line 46-49, "... static analysis... can work backwards... to identify the actual instruction that caused the event.")

As per claim 2, the rejection of claim 1 is incorporated; further Anderson et al. disclose **the**

ambiguity creating location is a branch target location (col. 23, line 50-55, "… the branches and not the actual target destinations, the information can be imprecise. In particular, merges in control flows can create ambiguities in identifying actual path…").

As per claim 3, the rejection of claim 2 is incorporated; further Anderson et al. disclose **the ambiguity creating location is bridged using branch history information** (col. 25, line 17-22, "For each instruction executed in the trace, work backwards to determine path segments until either:
(a) the global branch history bits are exhausted…").

As per claim 4, the rejection of claim 1 is incorporated; further Anderson et al. disclose **the ambiguity creating location is an entry point location** (col. 23, line 50-55, "… the branches and not the actual target destinations, the information can be imprecise. In particular, merges in control flows can create ambiguities in identifying actual path…" The examiner asserts that an entry point location creates merges in control flows, so it creates ambiguities.)

As per claim 5, the rejection of claim 1 is incorporated; further Anderson et al. disclose **the ambiguity creating location is one of: a jump target location; an indirect branch target location; a trap handler location; and an interrupt handler location** (col. 23, line 56-58, "asynchronous events that cause branched code to execute… such as interrupts… can pollute branch history…").

As per claim 6, the rejection of claim 1 is incorporated; further Anderson et al. disclose **the preceding operation corresponds to a load instruction** (; **and the execution event is a cache miss** (col. 12, line 31, "Events could include cache misses…").

As per claim 7, the rejection of claim 1 is incorporated; further Anderson et al. disclose **the preceding operation corresponds to a memory access instruction** (col. 9, line 49-50, "… execute memory access instructions…"); **the execution event is either a hit or a miss at a level in a memory hierarchy** (col. 12, line 31, "Events could include cache misses…").

As per claim 8, the rejection of claim 1 is incorporated; further Anderson et al. disclose **the execution event is either an overflow or an underflow of a hardware counter** (col. 5, line 38-40, "… sampling the program counter when event counter overflow…"; col. 6, line 62-65, "… hardware event-counters…").

As per claim 9, the rejection of claim 1 is incorporated; further Anderson et al. disclose **the execution event triggers either an overflow or an underflow of a hardware counter that is itself detected** (col. 12, line 31-35, "Events could include cache misses…"; col. 5, line 38-40, "… sampling the program counter when event counter overflow…"; col. 6, line 8, "Special-purpose hardware could count…").

As per claim 10, the rejection of claim 1 is incorporated; further Anderson et al. disclose **the latency includes that associated with delivery of a trap** (col. 9, line 59-65, "…some instructions may abort or be trapped. For example, the execution flow may change after an instruction is fetched, or an instruction
may suffer an exception trap. In these cases, the instruction and all subsequent instructions already in the pipeline are discarded and the speculative processing state is rolled back." The examiner asserts that the action to discard or rollback instructions takes time and inherently introduces latency.)

As per claim 11, the rejection of claim 1 is incorporated; further Anderson et al. disclose **the latency includes that associated with delivery of a counter overflow event signal** (col. 12, line 51-68, "Latency registers store timing information taken at check points… The checkpoints may differ from processor to processor depending on where an instruction might be stalled waiting for some event or resource. Each latency register counts the number of cycles an instruction spent between two checkpoints." The examiner interprets that one processor may use a counter overflow event as check point for latency.)

As per claim 12, the rejection of claim 1 is incorporated; further Anderson et al. disclose **the latency is associated with pipeline execution skid** (col. 12, line 56-58, "… latency regist4r counts the number of cycles… between two checkpoints.").

As per claim 13, the rejection of claim 1 is incorporated; further Anderson et al. disclose **the latency is associated with completion of in-flight operation** (col. 12, line 51-52, "Latency registers stores timing information… while a selected instruction is in flight".)

As per claim 14, the rejection of claim 1 is incorporated; further Anderson et al. disclose **embodied in a computer program product** (col. 8, line 22-30, "… instructions and data of software programs are stored in the memories… instructions are decoded for execution.")

As per claim 15, the rejection of claim 1 is incorporated; further Anderson et al. disclose **embodied in at least one of: a profiling tool; a code optimizer; and a runtime library** (col. 10, line 13-21, "This makes the profile record useful… profile-directed optimization…").

As per claim 19, Anderson et al. disclose a method of identifying operations associated with execution events of a processor, the method comprising:

from a point in an execution sequence of the operations of the processor, the point coinciding with an execution event, backtracking through the operations toward a particular operation that precedes the coinciding point by an predetermined latency; and associating the execution event with the particular operation unless the backtracking encounters an unresolved intervening target of a control transfer, wherein the predetermined latency is dependent upon the particular operation which is indicated

by the execution event. (col. 3, .3 line 46-49, "… static analysis… can work backwards… to identify the actual instruction that caused the event."; col. 7, line 7-9, "Latencies of instructions of the program are measured…".);

As per claim 20, the rejection of claim 19 is incorporated; further Anderson et al. disclose executing **the sequence of operations on the processor** (col. 26, line 61, "Machine code is executed…");

and detecting the execution event (col. 5, line 39-40, "… sampling the program counter when event counters overflow…").

As per claim 21, the rejection of claim 19 is incorporated; further Anderson et al. disclose **the operations are instructions executable on the processor** (col. 2 line 12-13, "… operation of a processor that can issue instructions…");

and the particular operation is a particular one of the instructions that triggers the execution event (col. 16, line 13-18, "As the instruction progresses… trigger signals… Ptrap… ").

As per claim 22, the rejection of claim 19 is incorporated; further Anderson et al. disclose **the operations correspond to instructions of program code.** (col. 2 line 12-13, "… operation of a processor that can issue instructions…").

As per claim 23, the rejection of claim 19 is incorporated; further Anderson et al. disclose **the execution event is an exception triggering execution of the particular operation** (col. 12, line 31-38, "Events could include…. And exception conditions…").

As per claim 24, the rejection of claim 19 is incorporated; further Anderson et al. disclose **the execution event is a cache miss** (col. 12, line 31, "Events could include cache misses…").

As per claim 25, the rejection of claim 19 is incorporated. It recites the same limitation as claim 10 and is rejected for the same reason set forth in connection with the rejection of claim 10 above.

As per claim 26, the rejection of claim 19 is incorporated; further Anderson et al. disclose **the execution event triggers a hardware event and the expected latency includes delivery of a signal associated therewith** (col. 12, line 51-68, "Latency registers store timing information taken at check points… The checkpoints may differ from processor to processor depending on where an instruction might be stalled waiting for some event or resource. Each latency register counts the number of cycles an instruction spent between two checkpoints." The examiner interprets that one processor may use a counter overflow event as check point for latency.)

As per claim 27, the rejection of claim 26 is incorporated; further Anderson et al. disclose **the hardware event is either underflow or overflow of a counter associated with the**

execution event (col. 5, line 38-40, "… sampling the program counter when event counter overflow…"; col. 6, line 62-65, "… hardware event-counters…").

As per claim 28, the rejection of claim 19 is incorporated; further Anderson et al. disclose **the execution event is either underflow or overflow of a counter** (col. 5, line 38-40, "… sampling the program counter when event counter overflow…").

As per claim 29, the rejection of claim 19 is incorporated; further Anderson et al. disclose **instances of intervening control transfer targets are identified in the execution sequence of operations to facilitate the backtracking** (col. 24, line 26-31, "… perform a backward analysis… can identify execution paths…").

As per claim 30, the rejection of claim 29 is incorporated; further Anderson et al. disclose **at least some of the instances of intervening control transfer targets are resolved using branch history information** (col. 25, line 17-22, "For each instruction executed in the trace, work backwards to determine path segments until either:
(a) the global branch history bits are exhausted…").

As per claim 31, the rejection of claim 19 is incorporated; further Anderson et al. disclose **control transfer target locations in the execution sequence are identified by a compiler** (col. 24, line 33-34, "The analysis can identify execution Paths…").

As per claim 32, the rejection of claim 19 is incorporated; further Anderson et al. disclose **the operations are instructions executable on the processor** (col. 2 line 12-13, "... operation of a processor that can issue instructions..."); **the particular operation is a particular one of the instructions that triggers the execution event**(col. 16, line 13-18, "As the instruction progresses... trigger signals... Ptrap... ").

As per claim 33, the rejection of claim 19 is incorporated; further Anderson et al. disclose **the operations correspond to instructions of program code**(col. 2 line 12-13, "... operation of a processor that can issue instructions...").

As per claim 34, the rejection of claim 19 is incorporated; further Anderson et al. disclose **preparing the execution sequence of the operations** (col. 8, line 22-25, "During operation of the system, instructions and data of software programs are stored in the memories. The instructions and data are generated conventionally using known compiler..." It is inherent that a compiler prepares the execution sequence of the operations.).

As per claim 35, the rejection of claim 34 is incorporated; further Anderson et al. disclose **preparation of the execution sequence includes identifying a location of the control transfer target therein** (col. 24, line 26-31, "... perform a backward analysis... can identify execution paths..." The examiner asserts that identification of paths inherently include control transfer target.).

As per claim 36, the rejection of claim 34 is incorporated; further Anderson et al. disclose **preparation of the execution sequence includes identifying a location of the particular operation therein** (col. 24, line 26-31, "… perform a backward analysis… can identify execution paths…" The examiner asserts that identification of paths inherently include identification of the location of the particular operation.)

As per claim 37, the rejection of claim 19 is incorporated; further Anderson et al. disclose **the particular operations include memory referencing instructions** (col. 12, line 13-15, "… the instruction is a memory access instruction…");

As per claim 38, the rejection of claim 37 is incorporated; further Anderson et al. disclose **the memory referencing instructions include one or more of loads, stores and prefetches** (col. 12, line 13-15, "… the instruction is a memory access instruction, such as a load…");

As per claim 39, Anderson et al. disclose a method of associating an execution characteristic of code with a particular operation thereof, the method comprising: **identifying at least first-type and second-type operations in the code** (col. 28, line 1-10, "looking for pairs where a load instruction is the first sample in the pair and where a use of the data from the load is the second sample in the pair.");

from a point in an execution sequence of the code that coincides with delayed detection of the execution characteristic, backtracking a predetermined displacement toward a candidate triggering operation of the first-type and associating the candidate triggering operation with the execution characteristic unless an unresolved intervening operation of the second-type is encountered, wherein the predetermined displacement is dependent upon the first-type operations which are indicated by the execution characteristic (col. 1, line 35-40, "… to monitor specific events during execution of a program…"; col. 3, line 46-49, "… static analysis… can work backwards… to identify the actual instruction that caused the event.")

As per claim 40, the rejection of claim 39 is incorporated; further Anderson et al. disclose **the first-type operations include memory access operations** (col. 28, line 1-10, "looking for pairs where a load instruction is the first sample in the pair and where a use of the data from the load is the second sample in the pair."; It is inherent that a load is a memory access operation; col. 12, line 13-15, "… the instruction is a memory access instruction… ").

As per claim 41, the rejection of claim 39 is incorporated; further Anderson et al. disclose **the second-type operations include operations that coincide with control transfer target locations in the code** (col. 15, line 23-29, "…control flow can branch into, or out of the group of fetched instructions")

As per claim 42, the rejection of claim 39 is incorporated; further Anderson et al. disclose **the execution characteristic involves memory access latency** (col. 28, line 2-3, "... measure the latency of the memory operations...").

As per claim 43, the rejection of claim 39 is incorporated; further Anderson et al. disclose **the execution characteristic includes a cache miss statistic** (col. 6, line 45-51, "... instruction cache miss and the latency incurred...").

As per claim 44, the rejection of claim 39 is incorporated; further Anderson et al. disclose **the detection delay includes a pipelined execution skid latency** (col. 12, line 51-53, "Latency registers store timing information... between various stages of the pipeline.").

As per claim 45, the rejection of claim 39 is incorporated; further Anderson et al. disclose **the second-type operations include operations that coincide with branch target locations in the code** (col. 5, line 16-19, "Some processors, such as the Intel Pentium, permit software to read the contents of the branch predictor's branch target... ");
at least some of the branch target locations are resolved using branch history information (col. 25, line 17-22, "For each instruction executed in the trace, work backwards to determine path segments until either:
(a) the global branch history bits are exhausted...").

As per claim 46, Anderson et al. disclose a method of preparing code, the method comprising:

preparing a tangible first executable instance of the code(col. 8, line 22-25, "During operation of the system, instructions and data of software programs are stored in the memories. The instructions and data are generated conventionally using known compiler…" It is inherent that a compiler prepares the execution sequence of the operations.),

the preparing identifying at least ambiguity creating locations therein (col. 24, line 33-34, "The analysis can identify execution Paths…");

executing the first executable instance and responsive to detection of an execution characteristic (col. 26, line 61, "Machine code is executed…"; col. 1, line 35-40, "… to monitor specific events during execution of a program…");

backtracking a predetermined displacement through the code to identify an associated operation thereof, wherein extent of the backtracking is limited at least by encountering of an unresolved intervening one of the identified ambiguity creating locations, wherein the predetermined displacement is dependent upon the associated operation which is indicated by the execution characteristic; (col. 3, line 46-49, "… static analysis… can work backwards… to identify the actual instruction that caused the event.");

and further preparing a tangible second executable instance of the code using the association between the associated operation and the execution characteristic (col. 7, line 3-5, "…a method is provided for optimizing a program by inserting memory prefetch operations in the program …" The examiner asserts that the code after inserting the prefetch operations is the second executable instance of the code.).

As per claim 47, the rejection of claim 46 is incorporated; further Anderson et al. disclose **the association between the associated operation and the execution characteristic is based on a set of additional detections and responsive backtracking** (col. 25, line 17-22, "For each instruction executed in the trace, work backwards to determine path segments until either: (a) the global branch history bits are exhausted…").

As per claim 48, the rejection of claim 46 is incorporated; further Anderson et al. disclose **the execution characteristic involves memory access latency** (col. 28, line 2-3, "… measure the latency of the memory operations…");
and **the preparation of the second executable instance includes insertion of prefetch operations into the code** (col. 7, line 3-5, "…a method is provided for optimizing a program by inserting memory prefetch operations in the program …").

As per claim 49, the rejection of claim 46 is incorporated; further Anderson et al. disclose **resolving at least some intervening ones of the identified ambiguity creating locations using branch history information** (col. 25, line 17-22, "For each instruction executed in the trace, work backwards to determine path segments until either:
(a) the global branch history bits are exhausted…").

As per claim 55, Anderson et al. disclose

an execution sequence of operations(col. 26, line 61, "Machine code is executed..."); **and one or more data sections that identify in the execution sequence at least ambiguity creating locations and target operations for use by one or both of a profiler and a optimizer**(col. 27, line 32-35, "... to collect statistical data about... experienced by various memory operations"; col. 15, line 60-61, "... memory location..."; col. 27, line 44-46, "... identify profitable locations ...for these memory operations..."; col. 7, line 14-15, "A program optimizer...") **, wherein the target operations occur at predetermined displacements from execution event unless at least one of the ambiguity creating locations is disposed therebetween, and wherein the predetermined displacements are dependent upon the target operations which are indication by the execution events** (col. 3, line 46-49, "... static analysis... can work backwards... to identify the actual instruction that caused the event.").

As per claim 56, the rejection of claim 55 is incorporated; further Anderson et al. disclose **the ambiguity creating locations include branch target locations** (col. 23, line 50-55, "... the branches and not the actual target destinations, the information can be imprecise. In particular, merges in control flows can create ambiguities in identifying actual path..."; col. 15, line 60-61, "... memory location...";).

As per claim 57, the rejection of claim 55 is incorporated; further Anderson et al. disclose **the**

target operations include memory referencing operations (col. 9, line 49-50, "… execute memory access instructions…").

As per claim 58, the rejection of claim 55 is incorporated; further Anderson et al. disclose **the one or more computer readable media are selected from the set of a disk, tape, magnetic, optical, semiconductor or electronic storage medium and a network, wireline, or wireless communications medium** (col. 8, line 23, "… software programs are stored in the memories…").

As per claim 59, Anderson et al. disclose

means for backtracking predetermined displacement, from a point coinciding with an execution event in an execution sequence of operations on a processor, through the execution sequence toward a particular operation thereof that precedes the coinciding point

and means for associating the execution event with the particular operation unless the backtracking encounters an intervening ambiguity creating location, wherein the predetermined displacement is depend upon the particular operation which is indicated by the execution event (col. 1, line 35-40, "… to monitor specific events during execution of a program…"; col. 3, line 46-49, "… static analysis… can work backwards… to identify the actual instruction that caused the event.").

As per claim 60, the rejection of claim 59 is incorporated; further Anderson et al. disclose **means for bridging at least some ambiguity creating locations** (col. 25, line 17-22,

"For each instruction executed in the trace, work backwards to determine path segments until either: (a) the global branch history bits are exhausted...").

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 16-18 rejected under 35 U.S.C. 103(a) as being unpatentable over Anderson et al. (US Patent No. 5,964,867) in view of Ronstrom (US PGPub. No. 2002/0010913).

As per claim 16, the rejection of claim 1 is incorporated.

Anderson et al. do not disclose padding operations.

However, Ronstrom discloses **a compiler that pads the code with one or more padding operations to absorb at least some instances of the latency** (p. 1, [0012], "... the compiler... generating a dummy instruction code for lowering cache miss penalty and inserting the same...").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the teaching of Ronstrom into the teaching **of** Anderson et al. to include padding dummy instruction. The modification would be obvious to one of ordinary

skill in the art to want to lower cache miss penalty (p. 1, [0012], line 12-13, "…lowering cache miss penalty…")

As per claim 17, the rejection of claim 16 is incorporated; further Ronstrom disclose **the padding operations are not themselves associated with the execution event** (p. 1, [0012], "… the compiler… generating a dummy instruction… and inserting the same…" It is inherent that a dummy operation does nothing but consumes processor cycles. So, dummy instructions are not associated with the execution events.).

As per claim 18, the rejection of claim 16 is incorporated; further Ronstrom disclose **the padding operations are not themselves ambiguity creating locations** (p. 1, [0012], "… the compiler… generating a dummy instruction… and inserting the same…" It is inherent that a dummy operation does nothing but consumes processor cycles. So, padding dummy instructions does not create ambiguity locations.)

4. Claims 50-54, and 61 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ronstrom (US PGPub. No. 2002/0010913) in view of Bala et al. ("Efficient Instruction Scheduling Using Finite State Automata").

As per claim 50, Ronstrom discloses a computer program product encoded in one or more computer readable media, the computer program product (p. 3, [0037], line 2-3, "… a computer program product…loadable into the memory..") comprising:

an execution sequence of operations (p. 4, [0063], line 3-4, "… a machine code running on a standard processor…");

and padding operations following at least some particular operations of the execution sequence, the padding operations providing an unambiguous skid region of the execution sequence(p. 1, [0012], line 9-13, "… detecting cache miss penalty… generating dummy instruction code for lowering cache miss penalty and inserting the same…");

determination of which of the at least some particular operation are associated with execution events (col. 3, line 46-49, "… static analysis… can work backwards… to identify the actual instruction that caused the event.").

Ronstrom does not disclose padding instructions between some particular operations and the ambiguity creating locations.

However, Bala et al. disclose padding instructions **between the at least some particular operations and subsequent ambiguity creating locations to provide a predetermined latenecy between the ambiguity locations and the at least some particular operations** (p. 47, 3rd para., line 6-10, "…insert instruction…when an instruction is speculatively scheduled above a branch…" The examiner asserts that a branch is an ambiguity creating location.).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the teachings of Bala et al. into the teachings of Ronstrom to include padding instructions between some particular operations and the ambiguity creating locations. The modification would be obvious to one of ordinary skill in the art to want to effectively detect hazards as suggested by Bala et al. (p. 47, left col., line 4-5, "…effectively reducing the problem of structural hazard detection…")

As per claim 51, the rejection of claim 50 is incorporated; further Ronstrom discloses **the particular operations include memory access operations** (p. 1, [0012], line 9-13, "… detecting cache miss penalty… generating dummy instruction code for lowering cache miss penalty and inserting the same…" The examiner asserts that operations related to cache miss are memory access operations.)

As per claim 52, the rejection of claim 50 is incorporated; further Ronstrom discloses **the padding operations include nops** (p. 1, [0012], line 9-13, "… detecting cache miss penalty… generating dummy instruction code for lowering cache miss penalty and inserting the same…" The examiner asserts that nops are dummy operations.)

As per claim 53, the rejection of claim 50 is incorporated; further Ronstrom discloses **the unambiguous skid region does not include an ambiguity creating location** (p. 1, [0012], line 9-13, "… detecting cache miss penalty… generating dummy instruction code for lowering cache miss penalty and inserting the same…" The examiner asserts that dummy operations do not introduce ambiguity, so it inherently does not include an ambiguity creating location.)

As per claim 54, the rejection of claim 50 is incorporated; further Ronstrom discloses **the one or more computer readable media are selected from the set of a disk, tape, magnetic,**

optical, semiconductor or electronic storage medium and a network, wireline, or wireless communications medium (p. 3, [0039], line 7-8, "… floppy discs and hard drives…" The examiner asserts that a hard drive is a magnetic storage medium.)

As per claim 61, it is the apparatus claim that recites the same limitation as claim 50 and is rejection for the same reason set forth for the rejection of claim 50.

Response to Arguments

In the remark,

Applicant argues:

1) Regarding the rejection of claim 1, Anderson does not teach or suggest identifying a preceding operation at a predetermined displacement, which is dependent upon the preceding operation and indicated by an execution event, from a detection point, unless an ambiguity creating location is disposed between the detection point and the preceding operation.

Examiner's response:

1) According to the limitation in claim 1, under the condition that "no ambiguity creating location is disposed therebetween", where examiner constructs the following segment of code with timing information.

T1 s1

T2 s2

....

T9 s9

T10 s10 --- detection point (trigger by s2 issued at T2)

This above short segment of code shows a sequence of statements s1-s10 that execute in the sequence s1 – s10 starting at time T1. Since there in no ambiguity creating location between s1 and s10, this code segment is supposed to run from s1 to s10 without any ambiguity. Assuming s2 issued at T2 later shows some potential problem and is detected by s10 at time T10 (per Anderson, col. 1, line 37, "...monitor specific events..."), since there is no ambiguity between s1 and s10, the displacement from the detection point to s2 is predetermined (col. 3, line 46-47, "...work backwards...to identify the actual instruction that caused the event..."). As a matter of fact, according to the Applicant's specification on page 10, [1030], line 10-11, "If no intervening ambiguity-creating location is encountered, association is straightforward.", it indicates the scenario presented by the examiner above.

Applicant argues:

2) Regarding claim 1 and 19, Anderson merely disclosed backtracking a fixed displacement from a detect point irrespective of the preceding operation.

Examiner's response:

2) Applicant's arguments filed 1/3/2006 have been fully considered but they are not persuasive. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references. The Applicant

provides no explanation as to what it means by a fixed displacement, how it is different from a predetermined displacement in the claim and why Anderson's teaching teaches a fixed displacement. The examiner's response in 1) should fully support the reason for rejection of claims 1 and 19.

Applicant argues:

3) Regarding claim 19, Anderson does not teach or suggest identifying a preceding operation at a predetermined displacement, which is dependent upon the preceding operation and indicated by an execution event, from a detection point, unless an ambiguity creating location is disposed between the detection point and the preceding operation.

Examiner's response:

3) Based on the scenario in the examiner's response 1), the latency between s10 and s2 is predetermined as the number of required CPU cycles (and subsequently time, or latency) to execute each instruction is known and predetermined.

Conclusion

THIS ACTION IS MADE FINAL. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37

CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Philip Wang whose telephone number is 571-272-5934. The examiner can normally be reached on Mon - Fri 8 - 44:00PM. Any inquiry of general nature or relating to the status of this application should be directed to the TC2100 Group receptionist: 571-272-2100.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

WEI ZHEN
SUPERVISORY PATENT EXAMINER